Project Name: MyPlaywrightProject

Purpose: Automate web application testing (functional tests) using Playwright with Java, integrating with TestNG for test execution and Allure for reporting.

Key Features: - Cross-browser automation (Chromium, Firefox, WebKit via Playwright) - Page Object Model (POM) for better maintainability - TestNG for test management, execution, and assertions - Screenshots captured on failure for debugging - Allure reporting for rich HTML test reports

Technology Stack: - Java: Programming language for test scripts - Playwright: Browser automation framework - TestNG: Test execution and management - Allure: HTML reports with screenshots and logs - Maven: Project management and dependency handling - Page Object Model (POM): Modular and maintainable test design

Project Structure: - src/test/java/base/BaseTest.java: Browser and Playwright setup - src/test/java/pages/LoginPage.java: Page Object Model for login page - src/test/java/tests/LoginTest.java: Test scenarios for login functionality - src/test/java/listeners/TestListener.java: Captures screenshots on failure and attaches to Allure - testng.xml: TestNG suite configuration with registered listeners - pom.xml: Maven dependencies and plugins - target/allure-results: Allure JSON results and screenshots - screenshots: Optional folder for local screenshot storage

Workflow: 1. TestNG reads testng.xml and executes LoginTest 2. BaseTest initializes browser, context, and page 3. Tests interact with web elements via LoginPage methods 4. TestListener captures screenshots on test failure 5. Allure listener generates JSON files for reporting 6. Browser and context are closed after each test

Advantages: - Cross-browser testing - Reusable and maintainable code via POM - Automatic screenshots on failure - Rich, interactive Allure reports - Maven integration for CI/CD pipelines

Interview Script (2-3 minutes ready-to-speak): "My project, MyPlaywrightProject, is a web automation framework built using Java and Playwright. It follows the Page Object Model design pattern to separate test logic from page interactions, making the framework maintainable and scalable. We use TestNG to manage and execute tests, and Allure for detailed HTML reports. I implemented a listener that captures screenshots on test failures and attaches them to Allure reports, which greatly improves debugging efficiency. The framework is Maven-based, so tests can be executed and reports generated using a single Maven command, making it CI/CD ready. Overall, this framework demonstrates modularity, maintainability, and detailed reporting, which are crucial for professional QA automation projects."